We wish to compute the laziest way to dial given n-digit number on a standard push button telephone (with 12 keys) using two fingers. We assume that the two fingers start out on the * and # keys, and that the effort required to move a finger from one button to another is proportional to the Euclidean distance between them (assume that the digit "1" is at position (0, 0), and the digit "5" at position (1,1)). Design an algorithm in python that computes the method of dialing that involves moving your fingers the smallest amount of total distance. Your solution should be as good as possible from a time complexity perspective.

Your solution should implement the following function interface:

def compute_laziest_path(telephone_number: str) -> Tuple[float, List[Tuple[str]]]:

telephone_number is a string of length n representing the desired n-digit number. The function is supposed to return a tuple, where the first element of the tuple corresponds to the smallest amount of Euclidean distance both fingers together have to move to dial the telephone number, and the second element of the tuple represents the path of both fingers. It is supposed to be a list, where the i-th element of the list corresponds to the finger positions (itself a tuple, with the first element being the left hand finger position, and the second element being the right hand finger position) after dialing the i-th digit of the telephone number.

For instance, if telephone_number = "110", the function should return the following output:

(4.0, [('*', '#'), ('1', '#'), ('1', '#'), ('1', '0')])

You can assume that the string telephone_number does not contain any invalid characters (ie only elements representing one of the 12 keys).

What is the time complexity and space complexity of your solution, if k corresponds to the number of keys on the keyboard, and n to the length of the telephone number?